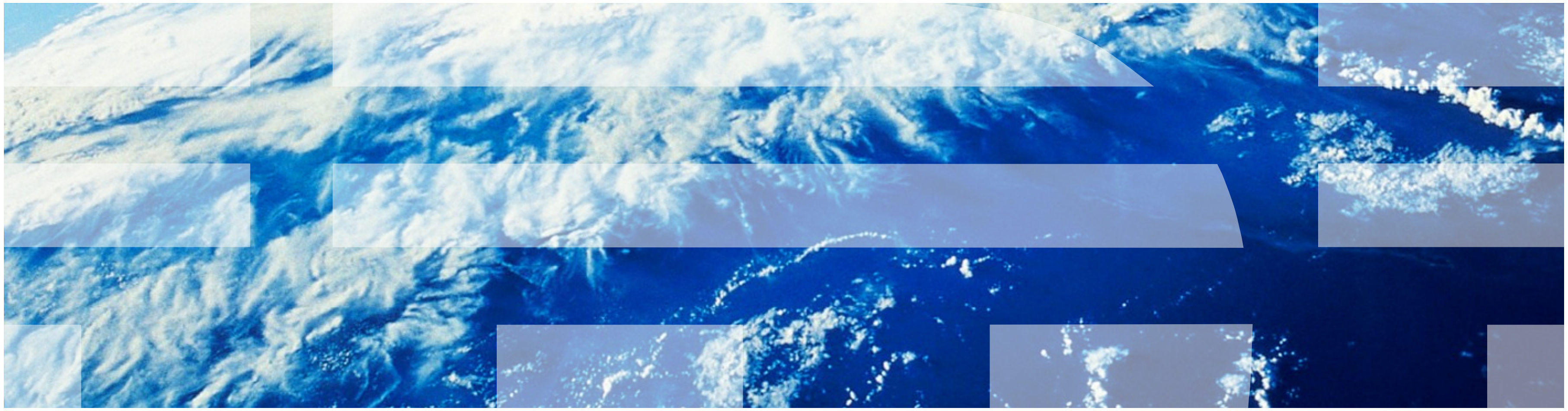
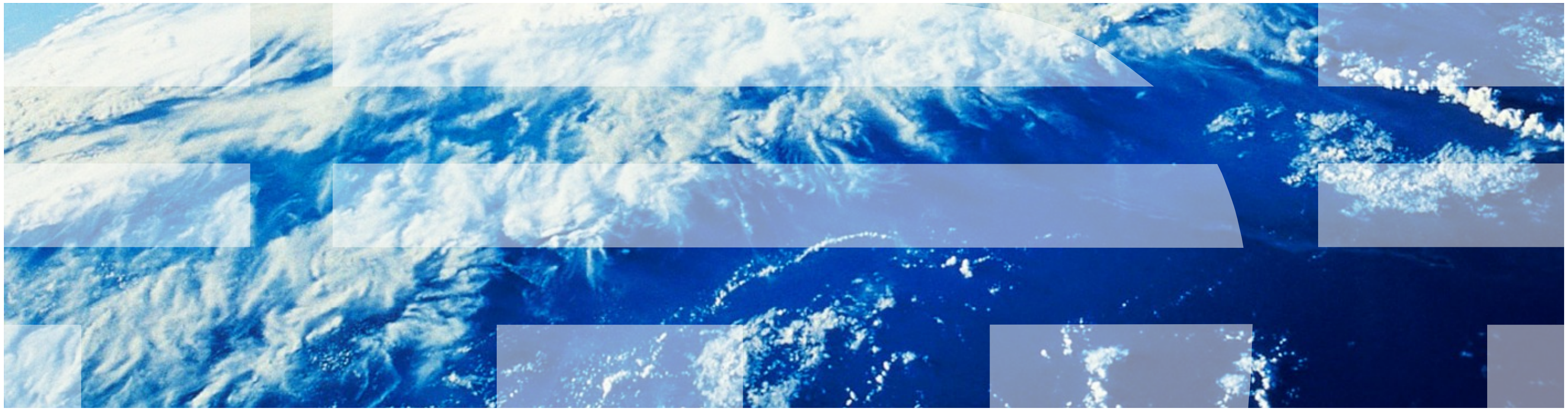
Lecture 1



Computer Systems for Data Science
Topic 1

Course Introduction

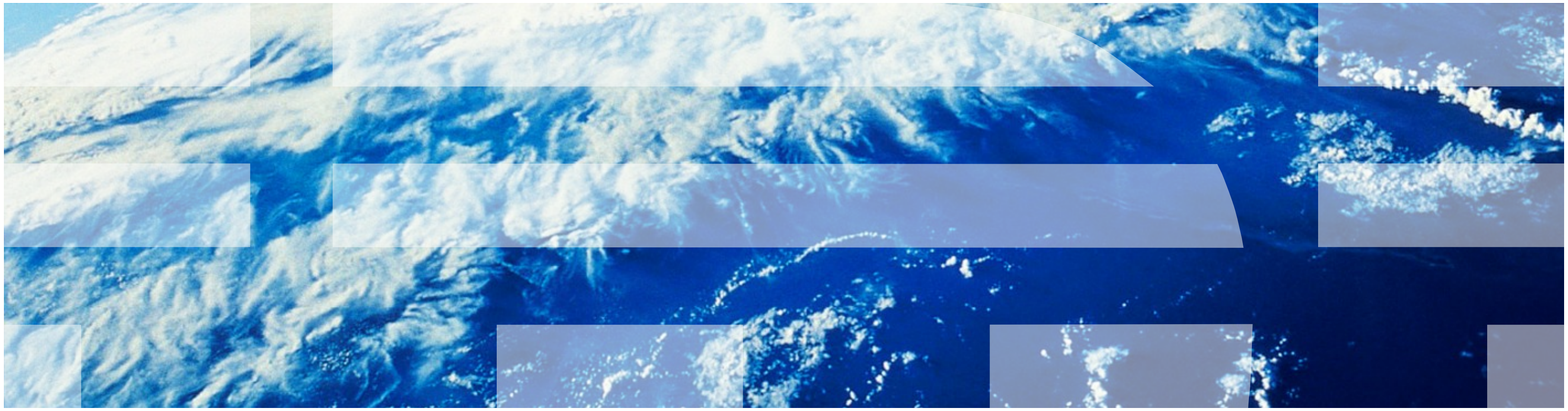
Systems concepts



Agenda

- Intro to instructors
- High-level overview
  - What is data science and big data?
  - Class goals and why should you care?
- Class logistics
  - How the class is going to work?
- Performance and systems rules of thumb
- Intro to datacenters

Who Are We?

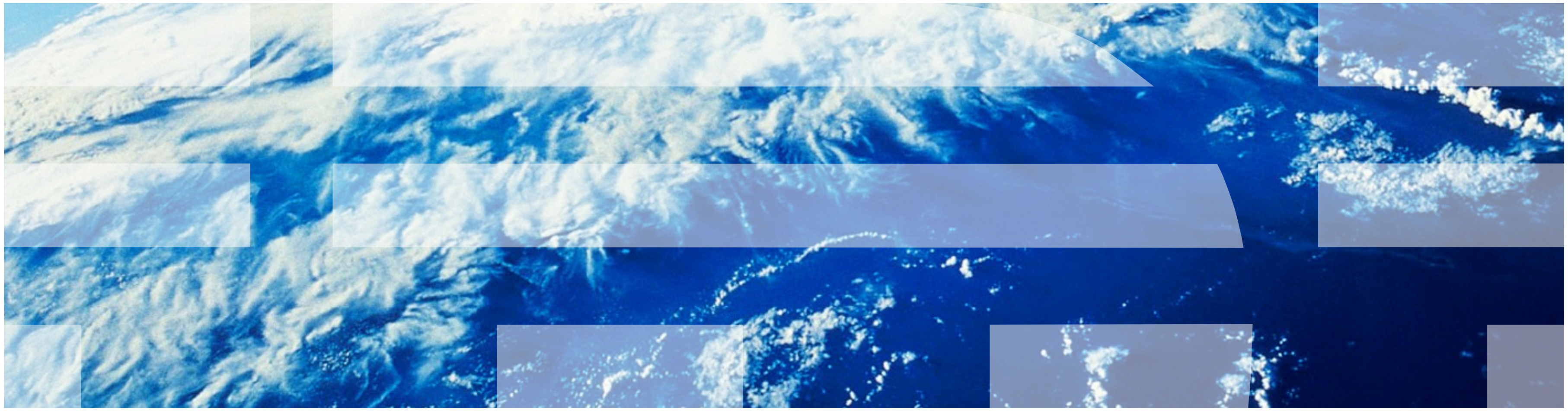


Course Instructors and TAs

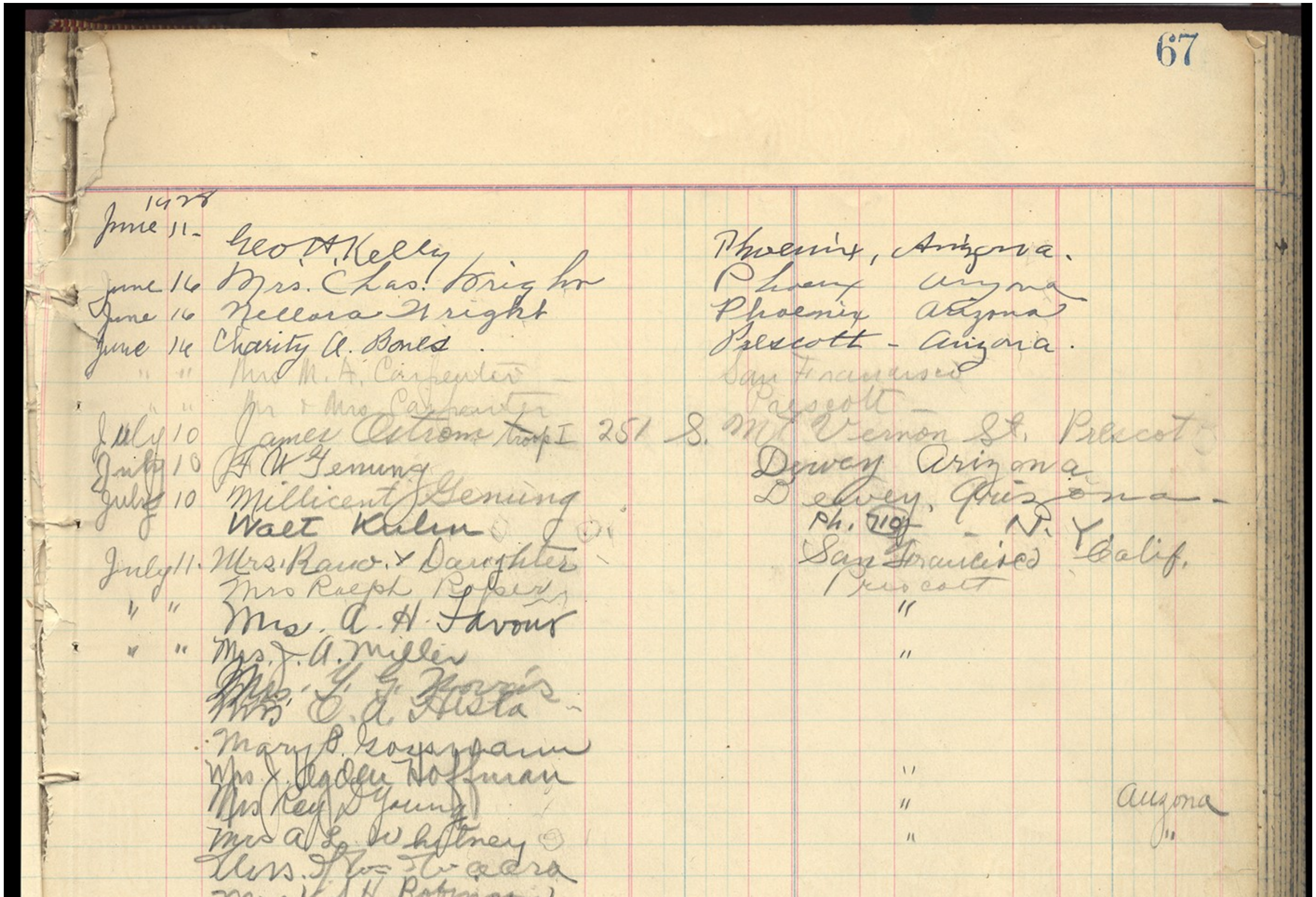
- Instructor: Asaf Cidon
  - New Columbia EE + CS professor
  - Research focus: infrastructure for big data, storage systems for big data
  - Before Columbia, founded Sookasa, cloud storage security company (acquired by Barracuda Networks)
  - Built big data systems and used them both in academia and in industry
  - Type of problems I've worked on:
    - Automatic classification of spear phishing
    - Anomaly detection of users within organization
    - Sensitive data exfiltration…

- Head TA: Hongyi Wang
- TAs: Yu Jian Wu, Ke Li, Mingen Pan, Qianrui Zhang
- All Tas have experience in databases and systems
  - Hongyi and Yu Jian helped create course homework

What is Data Science and Big Data?



This was a system for big data

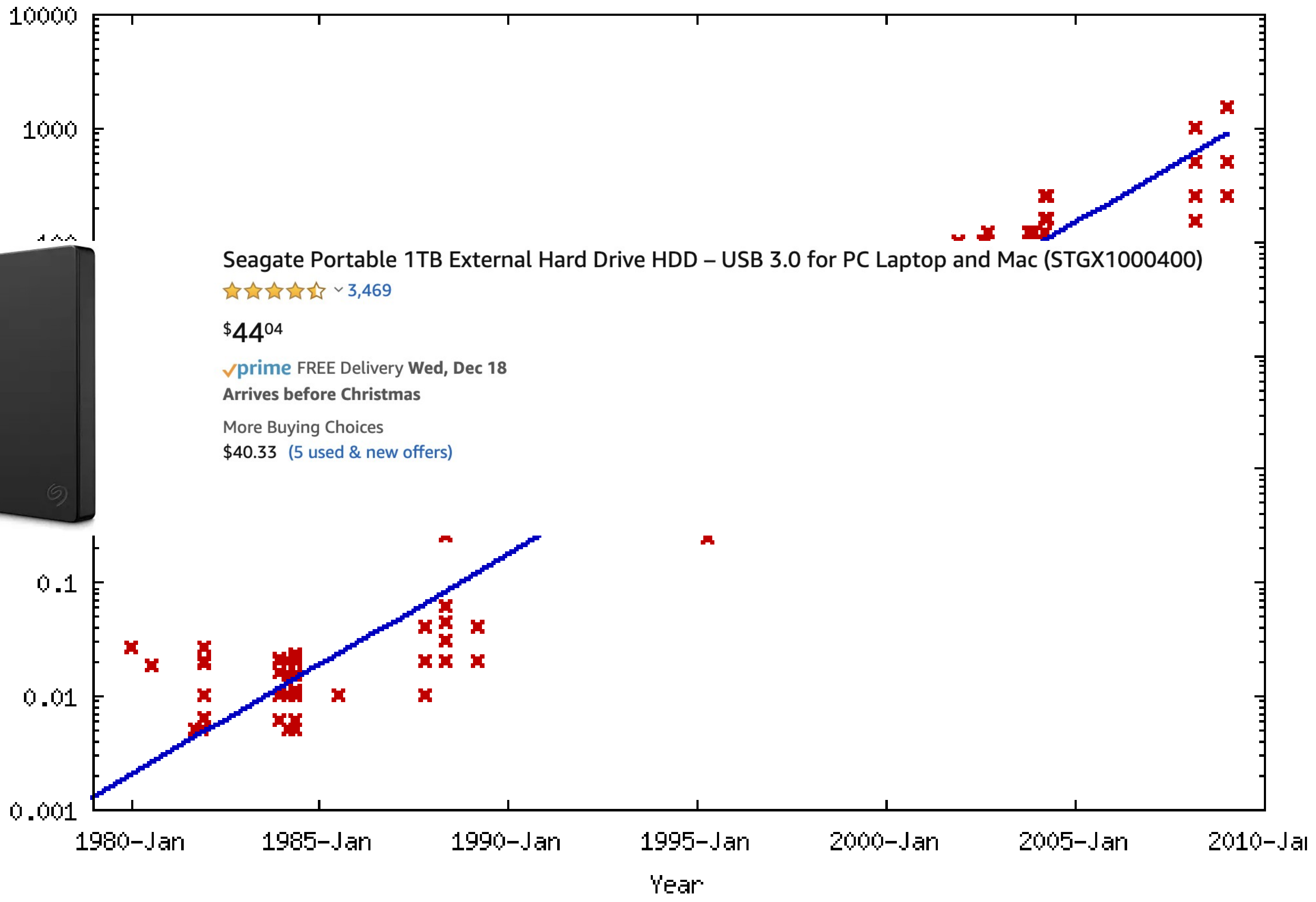


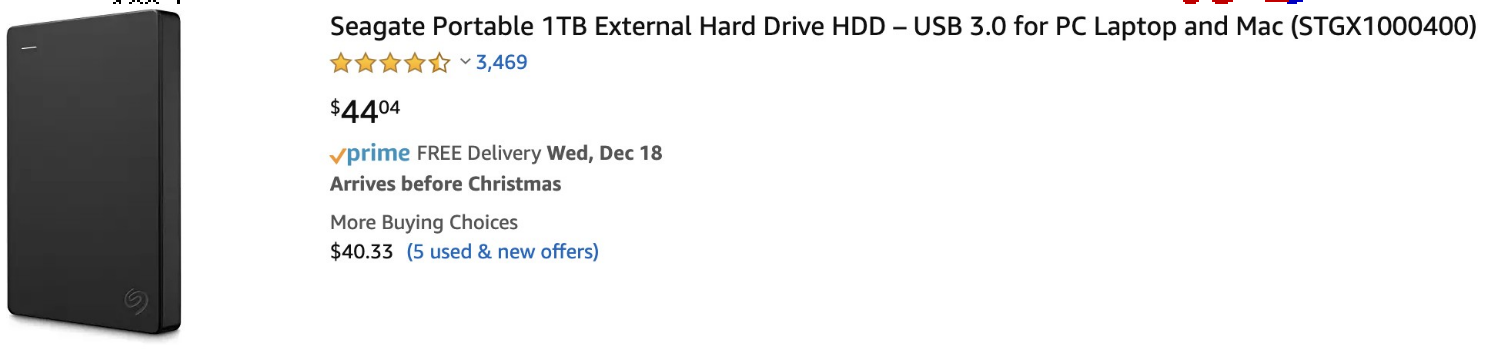
67

1928 June	11	Geo. A. Kelly	Phoenix, Arizona.	
June	16	Mrs. Chas. Wright	Phoenix Arizona	
June	16	Nellora Wright	Phoenix Arizona	
June	14	Charity A. Bones	Prescott - Arizona.	
"	"	Mrs M. A. Carpenter	San Francisco	
"	"	Mr & Mrs Carpenter	Prescott	
July	10	James Ostrom troop I	251 S. Mt Vernon St. Prescot	
July	10	A. W. Genung	Dewey Arizona	
July	10	Millicent J. Genung	Dewey, Arizona	
		Walt Kuhn	Ph. 710 - N. Y.	
July	11	Mrs. Rand & Daughter	San Francisco Calif.	
		Mrs Ralph Rogers	Prescott	
"	"	Mrs. A. H. Favour	"	
"	"	Mrs. J. A. Miller	"	
		Mrs. Y. G. Harris		
		Mrs. E. A. Hesla		
		Mary B. Gossmann		
		Mrs J. Ogden Hoffman	"	
		Mrs Roy D Young	"	Arizona
		Mrs A. L. Whitney	"	"
		Mrs. [illegible]		
		[illegible]		

Data science systems were expensive

Today: data is cheap




Seagate Portable 1TB External Hard Drive HDD – USB 3.0 for PC Laptop and Mac (STGX1000400)

3,469

$44.04

prime FREE Delivery **Wed, Dec 18**

Arrives before Christmas

More Buying Choices

$40.33 (5 used & new offers)

Where is data coming from?

- Physical devices

Where is data coming from?

- Physical devices
- Software logs

Where is data coming from?

- Physical devices
- Software logs
- Phones

Where is data coming from?

- Physical devices
- Software logs
- Phones
- GPS/Cars

Where is data coming from?

- Physical devices
- Software logs
- Phones
- GPS/Cars
- Internet of *Things*

Examples of data science

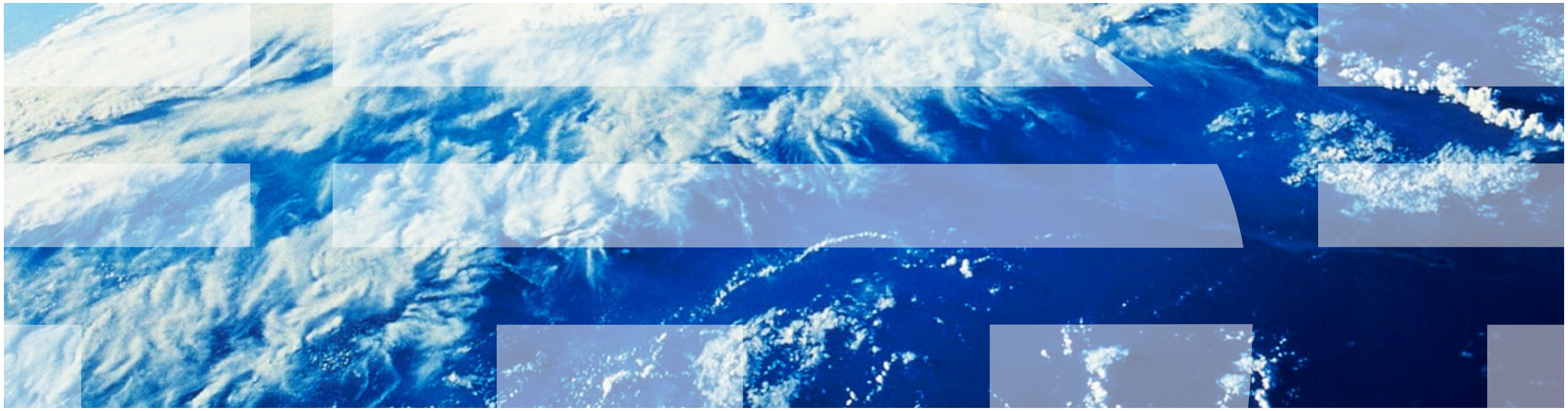
- What video should I recommend to this user to view next?
- Does this MRI image of a breast contain a tumor?
- Who is going to win the election?
- Which cities in the US will have high incidence of flu in 2 weeks?
- Is the object across from the car a pedestrian?

What is big data?

- “**Extremely large data sets** that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions” – Oxford Dictionary

- What’s an extremely large data set?
  - Fits on a single machine?
  - Fits on 10 machines?

Class Goals



Our focus in this class: **Computer Systems** for Data Science

- Questions we **will** answer in this class:
  - How are big data systems designed?
  - How to store the data?
  - How to query/analyze the data?
  - How do we ensure uptime/availability to the data?
  - How to ensure privacy/security/quality?
- Questions we **won't** answer in this class:
  - What algorithm should we use?
  - How to use ML for big data?
  - How do we explain/debug ML models?
  - How can data be visualized?
  - What are the statistical/mathematical foundations for data science?

Course Objectives

- **Graduate level course**
- **Broad overview of cloud systems that are used in data science**
  - **Database** related topics (DBMS, SQL, NoSQL)
  - **Computer systems** foundations (throughput vs. latency, scalability vs. performance)
  - **Distributed systems** for data scientists (sharding, fault tolerance)
- Throughout the class we will focus on how **commonly used and modern** cloud-based big data systems work (Spark, Tensorflow, Hadoop file system,…)
- The class will give a **broad and hopefully practical** introduction to these topics geared towards data scientists, but **does not replace** core CS/EE classes like OS, databases, distributed systems, security, architecture
- **You come from diverse backgrounds:** Some of the content will be repetitive for students who have taken the classes above
- **Required background**
  - Programming experience with Python or equivalent
  - Programming assignments 2-3 will be in pairs in Python, so we will make sure that all pairs have at least one student who knows Python

Course Administration and Grading

- **All materials, assignments, etc. posted on course website**
  - https://cs-w4121.github.io/
- **Announcement will be posted on Piazza, courseworks**
- **Lecture Materials**
  - Lecture slides
  - No textbook (new, fast moving field)
- **Homework, assignments, exams**
  - Programming assignment 1: BigQuery (20%), alone
  - Written assignment: systems and databases (10%), alone
  - Midterm (15%)
  - Programming assignment 2: Spark (20%), in pairs
  - Programming assignment 3: Tensorflow (10%), in pairs
  - Final exam (25%)

Programming Assignments

- 3 programming assignments
  - Assignment 1 is solo, assignments 2-3 are in pairs
- Programming assignments 2-3 are in Python
  - If you don't know Python
    - Pair with someone who does
    - A useful language to learn!
- All assignments in Google Cloud (GCP)
  - Goal: familiarize yourself with working in public cloud environment
    - AWS / Azure / GCP are similar
    - Many systems and deployment details are hidden / automated (but we won't ignore them!)
    - We will be focusing on systems-level problems, not on algorithms
  - We will provide GCP credits, if you run out contact us
    - If you reach $10 of credits or less, please contact Mingen: pan.mingen@columbia.edu
    - But be careful not to spend too many!
- Programming assignment goals
  - Assignment 1: BigQuery
    - Learning to use SQL on a big data set
  - Assignment 2: DataProc + Spark + HDFS + Streaming
    - Understanding Spark, Streaming systems concepts
  - Assignment 3: Tensorflow
    - Understanding Tensorflow and basic ML training systems problems

Tentative Contents and Syllabus

- Computer systems and performance rules of thumb
  - Latency vs. throughput
  - Amdahl's law
  - Back-of-the-envelope systems math
  - Performance bottlenecks
- Data centers
  - What is a data center?
  - Data center failures
  - Achieving reliability with smart software
- Databases
  - Relational model and SQL
  - SELECT, FROM, WHERE
  - GROUPBY
  - JOINs
  - Nested queries
  - Transactions
  - ACID
  - OLAP vs. OLTP, SQL vs. NoSQL
  - Indexing
  - Logging
  - **System highlight: BigQuery, MySQL**
- Storage and distributed file systems
  - Storage technologies primer
  - Distributed file systems
  - **System highlight: Hadoop File System (HDFS), amazon S3/Google Cloud Storage**

Tentative Contents and Syllabus

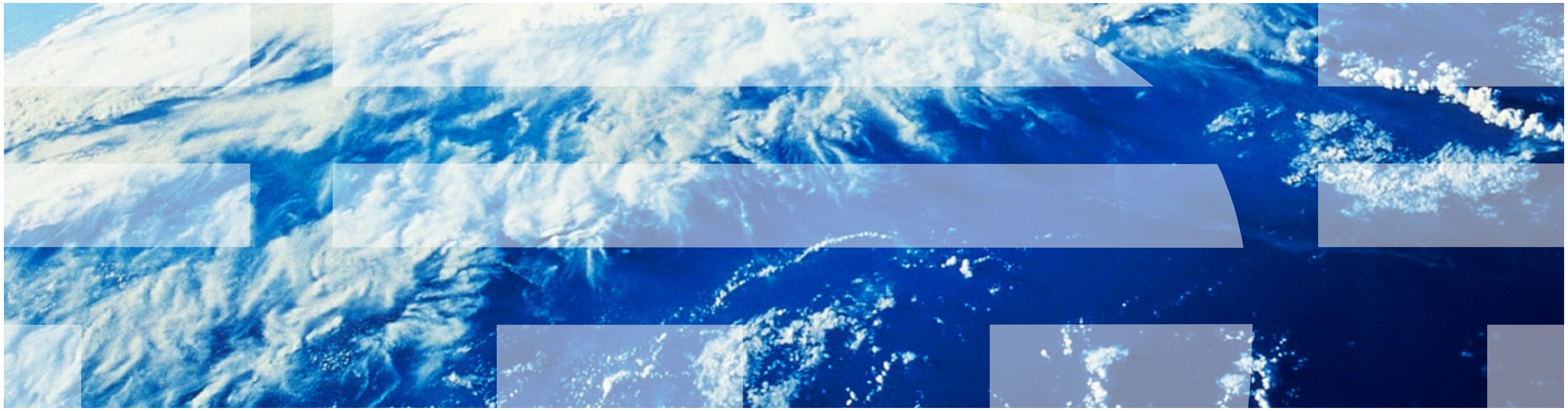
- Distributed systems
  - 2 Phase Commit
  - Locking
  - Sharding
  - Fault tolerance
  - Replication and consensus
- Mapreduce
  - Mapreduce computing model
  - Stragglers
  - Importance of 99th latency
  - Strategies to mitigate tail latency and increase availability
- Distributed analytics and streaming
  - Resilient Distributed Dataframes (RDD)
  - Fault tolerance in distributed analytics: lineage
  - Streaming computing model
  - **Systems highlight: Spark, Google Dataproc, Spark streaming**
- Caching
  - Performance benefits
  - When to use a cache? (hint: almost everywhere □)
  - Consistency and performance considerations
  - Eviction policies
  - **Systems highlight: Memcached and Redis**

Tentative Contents and Syllabus

- Tensorflow and pipelines
  - Tensorflow programming model
  - ML hardware trends
  - ML pipelines
  - Data validation and data quality
  - **Systems highlight: Tensorflow and Tensorflow Extended**
- Security and privacy
  - Security of big data systems
  - Privacy consideration
  - Data compliance and access control

Adapted from David Patterson and Kathryn McKinley

Performance Concepts and Rules of Thumb



Performance Evaluation

- Metric: something we measure
- Goal: evaluate how good/bad our computer system is performing
- Examples:
  - Power consumed by our database
  - Cost of running our web application
  - Average time it takes to render a user page
  - How many users can we support at the same time
- Metrics allow us to compare two computer systems

Tradeoff: latency vs. throughput

- Pizza delivery example
  - Do you want your pizza hot?
  - Do you want your pizza to be cheap?
- Why do these conflict?
- Two different strategies for pizza company
  - Often we have a requirement for both (I want my pizza to be delivered in X time as cheaply as possible)

- Latency = execution time for a single task
- Throughput = number of tasks per unit time

- A more relevant example:
  - Assuming cars drive at 65mph, so self driving car needs to recognize an object in 0.1 seconds
  - Object recognition system needs to process 1 million object recognition tasks every second

Latency vs. Throughput is often a trade off

Plane	DC to Paris	Speed	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
Concorde	3 hours	1350 mph	132	178,200

- Which plane has higher performance?

- Time to do the task (execution time)
  - **Latency**, execution time, response time
- Tasks per day, hour, week, sec (performance)
  - **Throughput**, bandwidth, operations per second

Definitions

- Performance is in units of things-per-second
  - Bigger is better

- Performance is in units of things-per-second
  - Bigger is better

- Response time of a system Y running Z

- Response time of a system Y running Z

  - $\text{performance}(Y) = \dfrac{1}{\mathit{execution\ time}\ (Z\ on\ Y)}$

- Throughput of system Y running many requests

- Throughput of system Y running many requests

  - $\text{performance}(Y) = \dfrac{\mathit{number\ of\ requests}}{\mathit{unit\ time}}$

- “System X is n times faster than Y” means:

- “System X is n times faster than Y” means:

  - $n = \dfrac{performance(X)}{performance(Y)}$

How do we improve performance?

- Suppose we have a database that processes two types of queries:
  - Query A finishes in 100 seconds
  - Query B finishes in 2 seconds
- We want better performance
  - Which query should we improve?
- The answer: it depends!

Speedup

- Make a change to the system
- Measure how much faster/slower it is

- Make a change to the system
- Measure how much faster/slower it is

- $Speedup = \frac{Execution\ time\ before\ change}{Execution\ time\ after\ change}$

Speedup when we know details about the change

- Performance improvement depends on:
  - How good is the enhancement? (factor S)
  - How often is it used? (factor p)
- Speedup due to enhancement E:
  - $Speedup(E) = \frac{Execution\ time\ without\ E}{Execution\ time\ with\ E} = \frac{Performance\ with\ E}{Performance\ without\ E}$
  - $ExTime_{new} = ExTime_{old} * \left[(1-p) + \frac{p}{S}\right]$
  - $Speedup(E) = \frac{ExTime_{old}}{ExTime_{new}} = \frac{1}{(1-p)+\frac{p}{S}}$

Amdahl’s law: example

- We built a new database that speeds up aggregate queries by 2x! **Hurray!**
- But.... only 10% of queries are aggregate queries
- $ExTime_{new} = ExTime_{old} * \left[(1-p) + \frac{p}{s}\right]$
- $ExTime_{new} = E$

Amdahl’s law in simple terms:
Make the common case fast!

- $Speedup_{total} =$ 0.95
- Amdahl’s law: speedup bounded by

- Amdahl’s law: speedup bounded by

- Even if aggregated queries could be completed in zero time, our **maximum** speedup would be:
- $\frac{1}{fraction\ of\ time\ not\ enhanced}$

- Even if aggregated queries could be completed in zero time, our **maximum** speedup would be:

- $Speedup_{optimal} = \frac{1}{0.9} = 1.111$

Useful back-of-the-envelope latency numbers (all rough estimates)

- Time measurements:
  - Nanosecond (ns): 1/1,000,000,000 second
  - Microsecond (us): 1/1,000,000 second
  - Millisecond (ms): 1/1000 second

- CPU cache access: 1ns
- Memory access: 100ns
- Read a small random object from a local flash drive: 50,000ns, 50us
- Read a small object within the same region in a data center: 100,000ns, 100us
- Run a SQL query on a flash database: 1,000,000ns, 1ms
- Read a small random object from magnetic disk: 10,000,000ns, 10ms
- Run a SQL query on a disk database: 20,000,000ns, 20ms
- Roundtrip time over the internet: 100,000,000ns, 100ms
- Bounded by the speed of light!

Database example

- Scenario:
  - A user application running in the cloud needs to read a small object
  - It first checks if the object is already saved locally, either in the CPU cache or in memory:
    - 10% chance it’s in the CPU cache
    - If not, 20% chance it’s in memory
  - If not saved locally, it fetches it from a database from within the center network

- Compute average latency:
  - 0.1 * cache latency + 0.9 * (0.2 * memory latency + 0.8 * (remote database latency))

    = 0.1ns + 18ns + 0.72 * not in memory latency
- Remote database latency = network latency + database latency = 1,100,000ns
- Total average latency = 792,018ns or 790us
- Total average latency ~= 0.72 * not in memory latency = 792,000ns
-
- **Amdahl’s law: focus on the most common path taken**

Disk vs. Flash, Cost vs. Performance

- Acme runs a flu prediction service
- They have an app that displays a graph on the geographic spread of the flu, which requires running a SQL query on their database stored in the cloud
- Acme is considering running their database on flash vs. magnetic disk
  - Flash is 2X more expensive, but 20X faster
- An Acme user study shows that users don't notice page loading times, as long as they are under 300,000,000ns (300ms)
- Reminder: Internet roundtrip (100ms), disk DB access (10ms), flash DB acccess (1ms), read within datacenter (100us)
- Scenario 1: To compute the graph, we only need a single database access in the cloud
  - Latency with flash database: 101ms
  - **Latency with disk database: 110ms**
- Scenario 2: The app requires getting an initial response from the database, then a user input, and then another database request
  - Latency with flash database: 202ms
  - **Latency with disk database: 220ms**
- Scenario 3: The app requires 20 sequential databases accesses within the cloud to compute the graph, and then it can return
  - **Latency with flash database: 120ms**
  - Latency with disk database: 300ms

Identifying performance bottlenecks

- My application is seeing an average latency of 200ms, where is the bottleneck?
- A few guiding questions:
    1. What systems does the web page need to access? Which networks does it need to traverse?
    2. Start from the most common case + highest latency
- Example:
    - Application needs to go through the Internet ~ 1 * 100ms
    - Hits a server that first checks if the request is saved on memory cache in the cloud ~ 0.2 * 100us
    - If not (80% of the time), goes over the network and accesses a single disk database ~ 0.8 * 10ms
- Guess 1: Internet slowdown
- Guess 2: database slowdown